Amendments to the Specification:

Kindly amend the third and fourth paragraphs on page 3 of the specification as follows:

Fig. 3 shows a physical layout for compressing a variable-sized field displaying [[the]] a variant use of offset slots.

Fig. 4 shows a physical layout for compressing a variable-sized ~~filed~~ field displaying [[the]] a variant use of field values for larger dictionaries.

Kindly amend the first paragraph of the Detailed Description on page 3 of the specification as follows:

Figure 1 is a flow diagram illustrating a routine for compressing large database tables in accordance with an embodiment of the invention. The data is received at step 101. The data received can be an arbitrary sequence of characters. The data received can consist of letters, for example an employee's name, title etc., the data can be numerical such as an employee's social security number, employee id etc. and the data can be combination of both letters and numbers. At step 102, the data is arranged in a mixed format layout, which is divided into fixed-sized fields (k), at step 103 and variable-sized fields (l) at step 104. An example of a physical layout of a mixed format is shown in Figure 2. In Figure 2, we consider a relation with k ~~fix-sized~~ fixed-sized fields and I variable-sized fields. The physical layout, 200, in mixed format, of this relation has k+l fixed fields, (k values and 1 field offsets) in the front of the record and l variable fields after. The sizes of the ~~fix-sized~~ fixed-sized fields and the order of all fields are stored in a data dictionary (not shown), along with such global (common to all records) information such as the types of each field, any integrity constraints, and so on. An example of the type of data or record in the fixed-sized field would be an employee's social security # since the ss# [[is]] always consists of 9 digits. An example of the type of data or record in the variable-sized field would be employees' name or address, which would vary in digits. Back to Figure 1, finally at step 105, the data in the

fixed-sized fields are compressed, and at step 106, the data in the variable sized fields are compressed. Various compression methods are well-known in the art. For example, a compression technique called Byte Pair Encoding (BPE) is presented by Philip Gage in "A New Algorithm for Data Compression - The C Users Journal, February, 1994". More detailed compression of the data in the fields is described below.

Kindly amend the paragraph beginning on page 5 of the specification as follows:

Next, we take a look at a variant interpretation of the fixed-sized field itself, as illustrated in Figure 5. Figure *5* shows a typical mixed format layout, 501, in which fixed-sized fields are overloaded to store field values, field offsets, or pointers into compression dictionaries. A fixed-sized field of uniform and small size is often not worth compressing, because the additional bits needed to code a variable field resulting from that might erase the gain of compression. However, sometimes there are fixed-sized fields that can use a smaller size except for a small fraction of large values. In this case, allowing exceptions to the fixed-sized format can achieve compression. An exception value for a fixed-sized field can be coded as an offset (stored in the fixed-sized slot), that points to an additional variable-sized field towards the end of the record. For example, as shown in Figure 5, an exceptionally large value [[F_1*]] F_1' for a fixed-sized field F_1 is stored as an extra variable-sized field. The fixed slot for F_1 is used to store the offset pointer to terminate [[F_1*]] F_1'.